An O(N log N) Expected Time Merge Heuristic for the Planar ETSP

Ritu Chadha and Donald C. S. Allison

TR 88-16

An O(N log N) expected time merge heuristic for the planar ETSP

Ritu Chadha and Donald C. S. Allison
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061.

Abstract

We discuss a new heuristic for solving the Euclidean Traveling Salesman Problem (ETSP). This heuristic is a convex hull-based method and makes use of the Delaunay triangulation of the set of cities to compute a tour for the given set of cities. We conjecture that the expected running time for this algorithm is O(N log N), implying that a new and faster ETSP heuristic is now available.

1. Introduction

In this paper, we discuss a new heuristic for solving the Euclidean Traveling Salesman Problem (ETSP). The traveling salesman problem may be stated as follows : if we have a set of N cities, and a salesman has to visit each of the N cities exactly once and then return to his starting point, what is the order in which he should visit the cities (called the **tour**) in order to minimize the total distance which he has to travel? In this problem, the cities are given as a set of points in the Euclidean plane, i.e. they are given in cartesian coordinates, and the distances between the cities are measured using the Euclidean measure of distance in the plane. It has been shown by Garey and Johnson [8] that this problem is NP-complete. Thus a lot of attention has been focused on heuristics which run in polynomial time and generate tours which are close to optimal. In general, there is a tradeoff between speed and quality; i.e. faster algorithms do not approximate the optimal tour as well as some costlier algorithms. Thus it is necessary to achieve a suitable balance between cost and quality.

The various approximation methods used for the ETSP can be classified into two broad categories : tour construction procedures and tour improvement procedures. Tour construction procedures start with a set of points representing the cities to be visited, and generate a tour of the cities which approximates the optimal tour. Tour improvement procedures take a given tour of a number of cities, and try to improve the approximation of that tour to the optimal tour (which is not generally known). In this paper, we will concentrate on tour construction procedures. We briefly describe some of the better-known heuristics which have been used for constructing Euclidean Traveling Salesman tours.

The **nearest neighbor** heuristic [Rosenkrantz, Stearns and Lewis] starts with any city as the first city of the tour, and chooses the city nearest to this one as the next city on the tour. This process is repeated until all the cities have been visited. The nearest neighbor heuristic has a worst-case time performance of $O(N^2)$. The **nearest insertion** heuristic [Rosenkrantz] starts with an arbitrary city as the initial subtour, finds a city which is the nearest to some point on the subtour, and inserts it between two adjacent cities i and j on the subtour so as to minimize $d_{ik} + d_{kj} - d_{ij}$, where d_{rs} is the distance between cities r and s. The **farthest insertion** heuristic operates in a similar fashion, except that at each step, the city chosen for insertion is the one farthest from any point on the subtour, and it is inserted between adjacent cities i, j already

on the subtour so that $d_{ik} + d_{kj} - d_{ij}$ is maximized. The nearest insertion and farthest insertion heuristics also have a worst-case time performance of $O(N^2)$.

The **Stewart Hull Heuristic** uses the convex hull of the set of points representing the cities to be visited as the initial subtour of the cities. Then, for each city k not yet contained in the subtour, it finds a pair of adjacent cities i, j already on the subtour such that $d_{ik} + d_{kj} - d_{ij}$ is minimal. After such a pair (i, j) is found for every city k not yet on the subtour, a triple (i*, j*, k*) is chosen from all the triples (i, j, k) above such that $d_{i^*k^*} + d_{k^*j^*} - d_{i^*j^*}$ is minimal. The city k* is then inserted into the subtour between cities i* and j*. The above procedure is repeated until all the cities have been inserted into the subtour. The Stewart Hull Heuristic has a worst-case time performance of $O(N^3)$, but empirical studies have shown that in practice, the expected time performance of this heuristic is $O(N^2 \log N)$.

From the extensive analytical and empirical studies in the literature about the above algorithms, it seems safe to say that in general, the performance of the Stewart Hull Heuristic has been found to be superior to all the other methods mentioned above (see [1, 2]. For this reason, in Section 4, we will compare the performance of our heuristic with the performance of the Stewart Hull Heuristic.

This paper is organized as follows. The next section describes the relationship between minimum spanning trees and Euclidean Traveling Salesman tours. Section 3 gives a brief overview of the definition of a Delaunay triangulation (which will be required in Section 4) and an asymptotically optimal divide-and-conquer algorithm for constructing the Delaunay triangulation of a set of points. Section 4 describes in detail a new heuristic, which we call Heuristic C, for finding a traveling salesman tour of a set of points in the Euclidean plane. In Section 5, we compare the performances of the Stewart Hull heuristic and Heuristic C on data sets of several sizes. Finally, we present the conclusions of this study in Section 6.

2. Minimum spanning trees

In this section, we describe how the length of the minimum spanning tree of a set of points can be used in estimating a lower bound for an optimal ETSP tour. Recall that a spanning tree

for a set of n cities is a set of n-1 edges which join all cities into a connected component. Using a well-known algorithm such as Kruskal's algorithm for minimum spanning trees [6], the minimum spanning tree for a set of N cities can be found in time $O(N^2)$. Now suppose that we know the optimal tour for a set of n cities. If we delete one edge from this tour, we will be left with a spanning tree whose length must be less than or equal to the length of the minimum spanning tree. Thus the length of the optimal traveling salesman tour must be strictly greater than the length of the minimum spanning tree, and so we have obtained the result that the length of the minimum spanning tree is a lower bound on the length of an optimal Euclidean traveling salesman tour. However, a tighter lower bound was obtained by Akl [7], based on Monte Carlo experimentation. He obtained a theoretical lower bound of

$$MST * 1.102$$

for a Euclidean traveling salesman tour, where

MST = length of the minimum spanning tree of the cities in the tour.

We shall use this result as a basis for estimating the quality of the tours produced by the heuristics which we will test.

3. Delaunay triangulation

In this section, we give a brief exposition of the problem of finding the Delaunay triangulation of a set of points. Let $V = \{v_1, v_2, ..., v_n\}$ be a set of distinct points in the Euclidean plane, such that all the points in V are not collinear and no four points in V are cocircular. For any two points $u = (x_1, y_1)$ and $v = (x_2, y_2)$ in V, define the distance d(u, v) to be $d(u, v) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Define the set

$$V(i) = \{x \mid d(x, v_i) \leq d(x, v_j), j = 1, 2, ..., N\}.$$

Then V(i) contains all the points which are closer to v_i than to any other point in the set V. The polygon V(i) is called the **Voronoi polygon** of the point v_i , and v_i is called the **growth center** of V(i). Voronoi polygons which share a common edge are called **Voronoi neighbors.**

Consider two points v_i and v_j , whose Voronoi polygons are Voronoi neighbors. This means that V(i) and V(j) share an edge in common. Now, every point on this common edge must be equidistant from both v_i and v_j, since it is the boundary separating the two sets of points which are closest to v_i and to v_j respectively. Thus the common edge e_1 between V(i) and V(j) is the perpendicular bisector of the line joining v_i and v_j (see Figure 1). Now consider one of the endpoints of this common edge e_1. There is an edge e_2 of the Voronoi polygon V(j) which is incident to an endpoint a of e_1. The edge e_2 must be an edge of a third Voronoi polygon V(k) of some point v_k. Hence every point on the edge e_2 is equidistant from points v_j and v_k. In particular, the point a (which lies on edge e_2) is equidistant from v_j and v_k. Also, a is equidistant from v_i and v_j (since it also lies on the edge e_1). Hence a is equidistant from all the three points v_i , v_j and v_k, and hence is the circumcenter of the triangle formed by connecting the points v_i , v_j and v_k.

Consider the aggregate of triangles formed by connecting the growth centers of Voronoi neighbors. This aggregate is called a **Delaunay triangulation** DT(V) of the set of points V, and any triangle in DT(V) is called a Delaunay triangle. From the above observations about the circumcenter of a triangle in a Delaunay triangulation of V, it is clear that given any Delaunay triangle, its circumcircle does not contain any other point of V in its interior. The converse of this statement is also true, i.e. a triangle T is a Delaunay triangle if and only if its circumcircle does not contain any other point of V in its interior (for proof see [3, 4]). In fact, this property is sometimes given as an alternative definition of the Delaunay triangulation.

We present below a divide-and-conquer technique for finding the Delaunay triangulation of a set V of points. This algorithm is due to Schachter & Lee [5] and is asymptotically optimal, running in time O(N log N).

First, the set V of N points is sorted and its elements are renamed so that

$i < j$ iff $v_i < v_j$, where $v_i < v_j$ means that either $x_i < x_j$ or $x_i = x_j$ and $y_i < y_j$, where $v_i = (x_i, y_i)$ and $v_j = (x_j, y_j)$.

V is then partitioned into two subsets V_L and V_R , where $V_L = \{v_1, v_2, \ldots, v_{\lfloor \frac{N}{2} \rfloor}\}$ and $V_R = \{v_{\lfloor \frac{N}{2} \rfloor + 1}, \ldots, v_N\}$. In effect, V_L is the set of the [N/2] leftmost points of the set V, and V_R contains the remaining points of V.

We recursively construct the Delaunay triangulations $DT(V_L)$ and $DT(V_R)$ of the two sets V_L and V_R respectively. The two triangulations $DT(V_L)$ and $DT(V_R)$ now have to be merged to form DT(V). For this, we need to construct the convex hull of the set of points V. We can do this recursively by finding the convex hulls $CH(V_L)$ and $CH(V_R)$ of V_L and V_R respectively and merging these two convex hulls. Now, while merging $CH(V_L)$ and $CH(V_R)$, we will get two new convex hull edges which are the lower and upper common tangents of the set of points V. These two common tangents will be in the Delaunay triangulation of V.

Next we merge the Delaunay triangulations of the two sets V_L and V_R , starting with the lower common tangent and zigzagging upward until the upper common tangent is reached. This is done as follows. Initially, we examine the points adjacent to the endpoints of the lower common tangent. Then, using the 'circle test ' (described below), we either connect :

(i) the left endpoint (in V_L) of the lower common tangent to a point adjacent to the right endpoint (in V_R) of the lower common tangent, or

(ii) the right endpoint (in V_R) of the lower common tangent to a point adjacent to the left endpoint (in V_L) of the lower common tangent.

The ' circle test ' will choose the point to be connected so that the circumcircle of the triangle formed by the connection of this point does not contain the other point being considered for insertion. The search for the point to connect begins with the points of V_L and V_R lying nearest to a vertical line separating V_L and V_R and alternately advances clockwise around the convex hull of V_L and counterclockwise around the convex hull of V_R .

This process is repeated with the newly found edge taking the place of the lower common tangent, and each succeeding edge taking the previous one's place. The process halts when the new edge found is the upper common tangent. As an illustration, consider the set of points shown in Figure 2. The points in V_L are shown as solid black points, and the points in V_R are shown as hollow points. The upper and lower tangents of the two sets of points and their convex hulls are also shown in the figure.

A proof of the correctness of this procedure may be found in [5]. The algorithm runs in time O(N log N).

4. Description of Heuristic C

We now describe the Heuristic C for approximating solutions to the ETSP. The method used here is based on the convex hull approach used by Stewart et. al. described earlier, but runs in O(N log N) expected time, as compared with the expected time of $O(N^2 \log N)$ for the Stewart Hull Heuristic. Heuristic C makes use of the Delaunay triangulation of the set S of N input points, merging it into the convex hull of S. As the merging proceeds, the initial Delaunay triangulation is dynamically reconfigured to ensure that every edge in the subtour has a viable insertion point. However, this scheme does not guarantee that all points will be added to produce the final tour. Points may remain inside a subtour which have edges connecting them only to non-consecutive subtour points. These points cannot be merged and are therefore added into the subtour by brute force using a cheapest insertion-no intersection rule.

We give a description below of the algorithm used to implement Heuristic C.

Let S = {V[1], V[2], ..., V[N]} be a set of N points in the Euclidean plane. Let DT(S) be the Delaunay triangulation of S and let CH(S) be the convex hull of S. The input to the algorithm consists of

(i) The cartesian coordinates of the points in the set S

(ii) A list of the indices (in V) of the points that make up each triangle in DT(S), and

(iii) An ordered list of the indices of the points lying on CH(S).

The algorithm is summarized below :

Step 1 : A single pass is made through the list of triangles and each edge and its (at most two) connection points are dynamically stored into a balanced binary tree B_1 (the key fields of B_1 are the endpoints of an edge in the triangulation).

Step 2 : The initial subtour ST is chosen to be CH(S).

Step 3 : For each edge (i, j) in the subtour ST a connection point k is found in B_1. The cost ratio to insert k into ST is computed using the formula

$$\text{cost ratio} = \frac{d(i,k) + d(k,j)}{d(i,j)}$$

and stored along with (i, j) and k into a priority queue Q.

Step 4 : The following steps are executed N - CH(S) times (until every point in S that is not on CH(S) is inserted into ST) :

Step 4.1 : A remove-queue operation is carried out on Q to find the smallest insertion ratio and its corresponding edge (i, j) and connection point k.

Step 4.2 : Each insertion is catalogued into another balanced binary tree B_2. The keys for B_2 are the endpoints of the edge (i, j) that is to be replaced by new edges (i, k) and (k, j) in ST.

Step 4.3 : We now update Q. B_1 is examined to find the connection points for (i, k) and (k, j). Insertion ratios are computed and stored into Q along with their edges and connection points.

Step 5 : Finally, a backtrack procedure can be used to fill in an array with the sequence of points comprising the final ETSP tour. It can be shown that this step makes at most n-3 accesses into B_2 and backtracks O(N) times.

It is a rather straightforward result that the merge algorithm can be implemented to run un O(N) time (worst-case performance). Also, the Delaunay triangulation of the set of points and the convex hull are computed in O(N log N) time. Thus the time complexity of the algorithm, assuming that no brute force insertions are required, is O(N log N). However, in the case where some points have to be inserted by brute force, the time complexity can go up to $O(N^2)$.

5. Testing and Results

As mentioned earlier, empirical evidence from the literature suggests that the Stewart Hull Heuristic is superior to the other approximate tour construction methods mentioned in this paper. Also, the Stewart Heuristic and Heuristic C are very similar in that their approach to constructing the Euclidean Traveling Salesman tour consists of starting with the convex hull of the given set of points as the initial subtour and repeatedly merging the remaining points into the subtour, until a complete tour is obtained. For this reason, we chose to use the

Stewart Hull Heuristic as a yardstick to measure and compare the quality of the tours produced by Heuristic C. Both the heuristics were coded in Pascal and run on the same set of data points, which were generated using a uniform random number generator. We used 10 sets of 25 points each, 10 sets of 50 points each, and 10 sets of 100 points each. We estimated the quality of the tours obtained by comparing them with the lower bound of

$$\text{MST} * 1.102,$$

where MST = length of the minimum spanning tree for the cities, as explained in Section 2. Thus we define the efficiency for a particular tour as

$$\text{efficiency} = \frac{\text{length of calculated tour}}{1.102 * \text{MST}}.$$

The results are shown in the tables below.

n = 25 points

Data set number	Stewart hull heuristic	Heuristic C
1.	1.0785	1.0997
2.	1.1976	1.2072
3.	1.1363	1.1091
4.	1.1738	1.2182
5.	1.1191	1.1755
6.	1.1138	1.0934
7.	1.1312	1.2229
8.	1.0840	1.1333
9.	1.1091	1.1093
10.	1.0856	1.0856

n = 50 points

Data set number	Stewart hull heuristic	Heuristic C
1.	1.1447	1.1932
2.	1.1040	1.1588
3.	1.1053	1.1491
4.	1.1298	1.1579
5.	1.1442	1.1455
6.	1.0969	1.1519
7.	1.0873	1.1316
8.	1.0839	1.1374
9.	1.1547	1.1787
10.	1.1052	1.0724

n = 100 points

Data set number	Stewart hull heuristic	Heuristic C
1.	1.1218	1.1389
2.	1.0756	1.1611
3.	1.0642	1.1241
4.	1.0927	1.1155
5.	1.1028	1.1046
6.	1.0790	1.1469
7.	1.0642	1.1241
8.	1.1374	1.1678
9.	1.1217	1.0829
10.	1.0906	1.0855

The means and standard deviations of the efficiencies shown above for n = 25, 50 and 100 are shown below :

	Stewart		Heuristic C	
	Mean	Standard deviation	Mean	Standard deviation
n = 25	1.12290	0.11655	1.14542	0.16482
n = 50	1.11560	0.07681	1.14765	0.09653
n = 100	1.09500	0.07645	1.12514	0.08745

Some of the tours obtained are depicted in Figures 3 and 4. We also ran the Stewart and the Heuristic C algorithms for problems 24 through 28, mentioned in [9]. The lengths of the tours obtained for these five 100-point problems are shown in the table below :

Problem number	Stewart	Heuristic C
24	22056	23526
25	22700	23507
26	21275	21233
27	21794	22740
28	22830	22995

From the results obtained above, we observe that the average efficiency for tours obtained using Heuristic C is approximately 2% worse than the average efficiency for tours obtained using the Stewart Hull heuristic for n = 25 and n = 100, and is 3% worse for n = 50. In the case of problems 24 through 28, the Stewart Hull heuristic performs slightly better than Heuristic C for problems 24, 25, 27 and 28, while Heuristic C performs marginally better for problem 26.

6. Conclusions

From the results in the previous section, we see that the performance of Heuristic C with respect to the quality of tours produced is only slightly worse than that of the Stewart Hull heuristic. Also, in all the above tests, no more than one point was ever added by brute force, and in many of the cases none of the points were added by brute force. Thus we conjecture that the expected running time of Heuristic C is O(N log N), which is considerably better than the expected running time of $O(N^2 \log N)$ for the Stewart Hull Heuristic. Thus in cases where a reasonably good tour is needed and where speed is important, it may be worthwhile to use Heuristic C in preference to other slower heuristics.

References

1. B. Golden, L. Bodin, T. Doyle, W. Stewart Jr., *"Approximate Traveling Salesman Algorithms"*, Operations Research 28 (1980), 694-711.

2. Mark Noga, *"Fast Geometric Algorithms"*, Ph.D. dissertation, Dept. of Computer Science, Virginia Tech (1984), 176-180.

3. D.T. Lee, *"Proximity and Reachability in the Plane"*, Ph.D. dissertation, Coordinated Science Lab. Report ACT-12, University of Illinois, Urbana, Illinois (1978).

4. M. I. Shamos, D. Hoey, *"Closest Point Problems"*, Proceedings of the 16th Annual Symposium on the Foundations of Computer Science (Oct. 1975), 151-162.

5. D. T. Lee, B. J. Schachter, *"Two algorithms for constructing a Delaunay triangulation"*, International Journal of Computer and Information Sciences 9 (1980).

6. J. A. Bondy, U. S. R. Murty, *"Graph Theory with applications"*, North-Holland, New York, 36-40.

7. S. G. Akl, *"An analysis of various aspects of the traveling salesman problem"*, Ph.D. thesis, School of Computer Science, McGill University, Montreal, Quebec, Canada (March 1978).

8. Michael R. Garey, David. S. Johnson, *"Computers and intractability : A guide to the theory of NP-completeness "*, W. H. Freeman & Co., New York 1979.

9. Patrick Krolak, Wayne Felts, *"A Man-Machine approach toward solving the Traveling Salesman Problem"*, Communications of the ACM 14 (May 1971), 327-334.

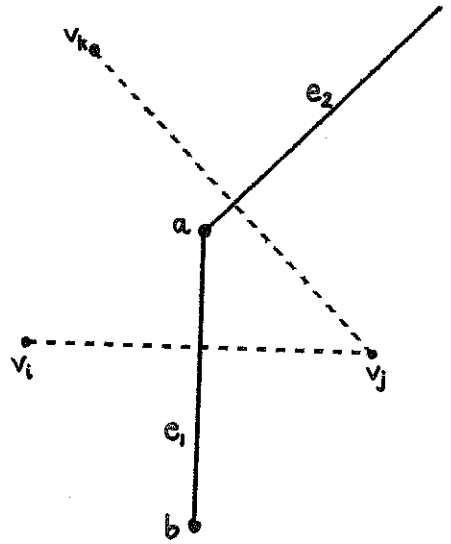

Figure 1

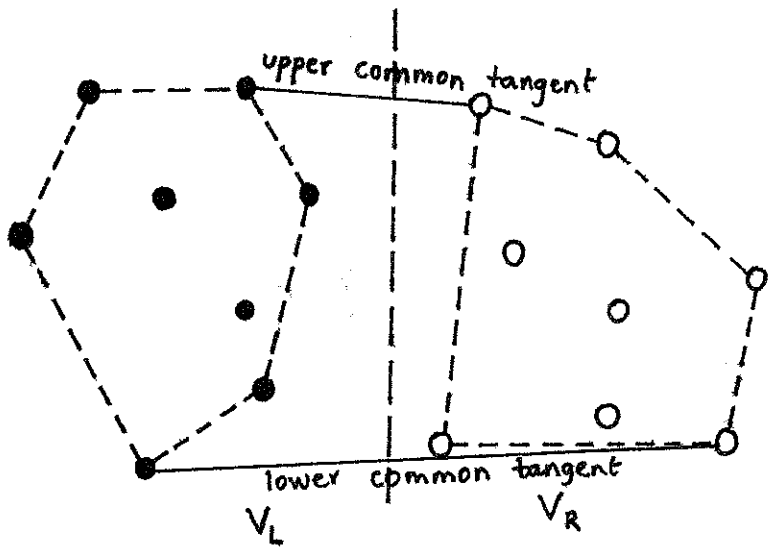

Figure 2

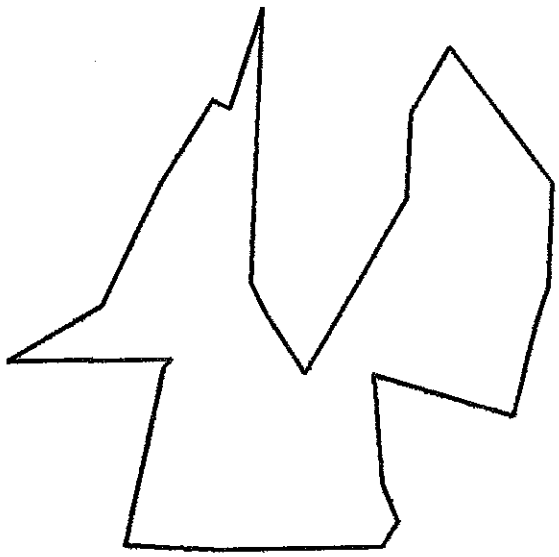

Figure 3. n=25, efficiency =1.1755 (Heuristic C)

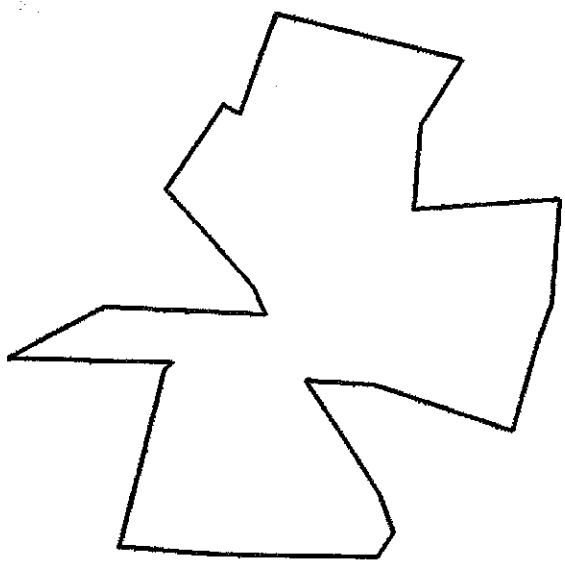

Figure 4 n=25, efficiency =1.1191 (Stewart)